UNITED STATES PATENT APPLICATION

for

SYSTEM AND METHOD FOR EFFICIENT TRANSMISSION AND DISPLAY OF IMAGE DETAILS BY RE-USAGE OF COMPRESSED DATA

Inventor:

Charles K. Chui

prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(408) 720-8598

File No.: 03971.P022

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL672748182US
Date of Deposit: March 13, 2001
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Mara E. Brown
(Typed or printed name of person mailing paper or fee)

Mara E Brown
(Signature of person mailing paper or fee)

3/13/01
(Date signed)

SYSTEM AND METHOD FOR EFFICIENT TRANSMISSION AND DISPLAY OF IMAGE DETAILS BY RE-USAGE OF COMPRESSED DATA

FIELD OF THE INVENTION

[0001] The present invention relates generally to the processing, compression, communication and storage of images in computer systems, personal digital assistants, cellular phones, digital cameras and other devices, and particularly to an image management system and method in which digitally encoded images can be viewed, in full or partially, in any specified window size and at a number of resolutions.

BACKGROUND OF THE INVENTION

[0002] An image may be stored at a number of resolution levels. The encoded image data for a lower resolution level is smaller, and thus takes less bandwidth to communicate and less memory to store than the data for a higher resolution level.

[0003] It is well known that wavelet compression of images automatically generates several resolution levels. In particular, if N "layers" of wavelet transforms are applied to an image, then N+1 resolution levels of data are generated, with the last LL subband of data comprising the lowest resolution level and all the subbands of data together forming the highest resolution

level. For convenience, the "layers" of wavelet transforms will sometimes be called "levels". Each of these resolution levels differs from its neighbors by a factor of two in each spatial dimension. These resolution levels are labeled herein as Level 0 for the lowest, thumbnail level to Level N for the highest resolution level, which is the resolution of the final or base image.

[0004] When using conventional as well as most proprietary data compression and encoding methods, the quantity of data in the N levels generated by wavelet compression tends to decrease more or less, depending on the quantization step, in a geometric progression. For instance, the quantity of data for resolution Level 0 is typically about 80% of the quantity of data for resolution Level 1, whereas ideally it should above 25% of the quantity of data for resolution Level 1. As a result, the data for Level 0, for example, contains significantly more data than is needed to display the Level 0 image. Alternately stated, the data for Level 0 gives unnecessarily high quality for the low resolution display at Level 0, and therefore gives less compression than could potentially be obtained by providing only the information needed for displaying the image at the Level 0 resolution level.

[0005] The low resolution image data coefficients are quantized for full resolution display, not for low resolution display, because these data coefficients are used not only for generating a low resolution representation of the image, but are also used when generating the higher resolution representations of the image.

[0006] It is well known in the prior art that digital images can be processed a portion at a time, instead of all at once, thereby reducing memory requirements. For instance, the DCT transform used for tile-based JPEG compression and encoding of images is traditionally used on blocks of 8x8 pixels. The restriction to tiles is necessary because the JPEG file format also depends on DPCM across the 8x8 pixel blocks. The file and multi-level DCT JPEG is governed by the Flashpix image file format. For more information, see Flashpix 1.0 Specification from the Digital Imaging Group at the world wide web site digital.imagery.org.

SUMMARY OF THE INVENTION

[0007] A method and apparatus for displaying images. In one embodiment, the method comprises displaying a first image at a first resolution level, identifying a location in the first image, and generating a second image for display at a second resolution level different than the first resolution level in response to user input via a user input mechanism. The second image represents a portion of the first image at a different second resolution level, which is dependant on the number of times the user input mechanism is utilized or activated (e.g., the number of mouse clicks made with a mouse).

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0009] **Figure 1** is a flow diagram of one embodiment of a process for generating multiple resolutions of an image.

[0010] **Figure 2** is a graphical illustration of the multi-resolution technique described herein.

[0011] **Figure 3A-C** is a graphical illustration of data re-use during panning.

[0012] **Figure 4** is a block diagram of a distributed computer system, including a web server and a number of client computers, for distributing multi-resolution images to the client computers.

[0013] **Figure 5** is a block diagram of a computer system in accordance with an embodiment of the present invention.

[0014] Figure 6A schematically depicts the process of transforming a raw image into a transform image array and compressing the transform image array into a compressed image file.

[0015] Figure 6B depicts a mapping of spatial frequency subbands to NQS subbands used for encoding transform coefficients.

[0016] Figure 7 is a conceptual representation of the encoded data that represents an image, organized to facilitate multi-resolution regeneration of the image (i.e., at multiple resolution levels).

[0017] Figure 8A, 8B, 8C, 8D and 8E depict image storage data structures.

[0018] Figure 9 is a high level flow chart of an image processing process to which the present invention can be applied

[0019] Figure 10A, 10B and10C graphically depict a forward and inverse wavelet-like data transformation procedure.

[0020] Figure 11 depicts the spatial frequency subbands of wavelet coefficients generated by applying multiple layers of a decomposition wavelet or wavelet-like transform to an array of image data.

[0021] Figure 12 depicts a flow chart of a block classification method for selecting a set of quantization divisors for a block of an image.

[0022] Figures 13A and 13B depict a flow chart of a procedure for encoding the transform coefficients for a block of an image.

[0023] Figure 14A, 14B and 14C depict a method of encoding values, called MaxbitDepth values in a preferred embodiment, which represent the number of bits required to encode the transform coefficients in each block and subblock of an encoded image.

[0024] Figure 15 is a high level flow chart of a compressed image reconstruction process to which the present invention can be applied.

[0025] Figure 16A and 16B depict a flow chart of a procedure for decoding the transform coefficients for an image and for reconstructing an image from the coefficients.

[0026] Figure 17 is a block diagram of a digital camera in which one or more aspects of the present invention are implemented.

[0027] Figure 18 is a conceptual flow chart of a client computer downloading a thumbnail image, then zooming in on the image, and then panning to a new part of the image.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

[0028] A method and apparatus for displaying images. In one embodiment, the method comprises displaying a first image at a first resolution level, identifying a location in the first image, and generating a second image for display at a second resolution level different than the first resolution level in response to user input via a user input mechanism. The second image represents a portion of the first image at a different second resolution level, which is dependant on the number of times the use input mechanism is utilized or activated. Such activation may include mouse clicks or other operations such as different keys on the keyboard of a personal computer (PC), different buttons on a personal digital assistant (PDA) or cellular phone, or touches on the display screen, when the zoom function is selected.

[0029] In the following description, numerous details are set forth to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

[0030] Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0031] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or

"displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0032] The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0033] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the

teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0034] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

Overview

[0035] Figure 1 is a flow diagram of one embodiment of a process for displaying an image. The process is performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as is

run on a general purpose computer system or dedicated machine), or a combination of both.

[0036] Referring to Figure 1, the process begins with processing logic displaying a first image at one resolution level (processing block 101). In one embodiment, the first image may be a thumbnail image. In order to display the first image, processing logic of a client computer system may have to download the corresponding image data from a server or other system across a network via a network connection. The image data may be in compressed format when retrieved. In such a case, the client retrieving the image data in compressed format includes processing logic for decompressing the image data.

[0037] Using the displayed image, processing logic identifies a location in the first image (processing block 102). In one embodiment, the location may be identified by positioning a cursor over the location in the image. The cursor may be positioned by a user selecting a particular portion of the first image to be shown at another resolution. For example, a user may wish to see a zoomed-in version of a portion of an image. The user is able to obtain a number of images representing zoomed-in versions of portions of an

image. In an alternative embodiment, the user may wish to pan the image, as opposed to zooming-into/out of and image.

[0038] In the case of using a cursor, the user may position the cursor with a mouse or some other well-known cursor control device (e.g., control keys, trackball, trackpad, etc.) in a computer system. The identification of the location of a curser, such as by, for example, x,y coordinates on a display and the identification thereafter of the location in an image being displayed under the cursor is well-known in the art.

[0039] Next, processing logic generates a second image for display at a second resolution level, where the second resolution level is selected by a user based on user input (processing block 103). The second image comprises a portion of the first image being displayed (over which the cursor is located) with that portion displayed at a resolution level that is different than the resolution level of the first image. The resolution level of the second image is selected by the user based on the number of times the user activates or engages a user input mechanism (e.g., the number of mouse clicks the user makes). For example, if the user selects a location and performs one mouse click, the second image may be displayed at a resolution level greater than that of the first image; if the user selects a

location and performs two mouse click, the second image may be displayed at a resolution level greater than that which is displayed after a single mouse click; etc.

[0040] Other operations may also be used to provide user input to trigger presentation of an image at a higher resolution (i.e., when the zoom function is selected). For example, selection of different keys on a keyboard of a personal computer (PC), depressing different buttons on a personal digital assistant (PDA) or cellular phone, or touches on a display screen. These user input mechanisms are all activated when used.

[0041] Figure 2 illustrates a graphical overview of the teachings of the present invention. Referring to Figure 2, an image 200 is displayed to the user. In one embodiment, image 200 is a thumbnail image. The user selects a particular location on image 200 by clicking a mouse on image 200 at the desired location 250. The resolution level of the image that is displayed is dependant on the number of mouse click that the user does. For example, if the user clicks on location 250 of image 200 once, image 210 is displayed. If the user clicks on location 250 of image 200 twice, image 220 is displayed. If the user clicks on location 250 of image 200 three times, image 220 is displayed. In this manner, when a user wants to see a certain point of

interest from the thumbnail (such as, for example, a section of the Golden Gate Bridge) or other image, the user simply positions the cursor over the certain point in the image and clicks a predetermined number of times to obtain the resolution they desire.

[0042] Utilization of a user input mechanism may also be used to control other operations, such as panning of an image, rotation of an image, etc. Figures 3A-C illustrate use of the data reuse technique described herein for panning. Figure 3A illustrates a thumbnail image 302 being displayed in display window 301. Once the user activates a user input mechanism (e.g., single clicks a mouse), a new image 303 is created using data of thumbnail 302 and additional data (e.g., downloaded data) and is larger than display window 301. A portion of new image 303 shown in display window 301 is data 310. When the user performs another user input mechanism activation (e.g. another mouse click), new image 304 is created using image 303 and additional data (e.g., downloaded data). As a user pans to the left, the image displayed in the display window 301 moves to the right. The image data, from the lower resolution levels, that remains in the display window is re-used to generate the image before panning is used to create the new image

along with data 310′ that is part of the image but outside the display window.

[0043] Data Re-use

[0044] In one embodiment, all of pictures that are obtained from the predetermined number of mouse clicks or other operations are of the same high quality, and the downloading (from the internet, for example) and time to display the new image is reduced, or may be even minimal, and the display image appears progressively (i.e., continuously) from "blur" to "clear" due to data re-use and efficient data management as is described in more detail below.

[0045] In one embodiment, when the first detailed picture is generated (for example, after one mouse click), the data in the thumbnail (or Level 0) is reused with additional downloaded data to create the image (or Level 1). Similarly, the Level 1 data is reused to generate the image in Level 2, etc. In other words, when a user clicks on a location a number of times (e.g., 1, 2, 3, etc.) (or utilizes another type of user input mechanism a number of times), processing logic in the system, which may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both,

identifies the location on the image being selected by the user, determines if it has the data to generate the new image locally, and, if not, sends a request to a server over a network or other communication medium to obtain the additional data needed to create the new image. The higher resolution image may be requested in response to a user indicating that a zoomed image is desired or desires some other operation to be performed in the image (e.g., panning). The data may be stored on the server or client side memory and subsequently sent in a compressed format. In such a case, the processing logic is responsible for decompressing the data as well.

[0046] In one embodiment, if the image is to be displayed in a viewing window of a particular size, then the processing logic downloads only the additional image data that is necessary to create an image of the size needed to display the image in the viewing window.

[0047] It should be noted that there is no limit for the design of the number of resolutions when using a compression scheme that is continuously scalable.

[0048] Exemplary Embodiments

[0049] In one embodiment, the techniques described herein are implemented as a viewer that enables a user to display images at multiple levels of detail. Such a viewer may supported using an image file and compression technology described in more detail below. Although at least one image file and compression technology are described herein, it would be apparent to those skilled in the art to employ other image file structures and/or different compression technologies.

[0050] In one embodiment, the viewer enables access and manipulation of an image to provide various sizes through the use of a browser. The images may be stored on the Internet and other resource accessible over a network via a network connection. The browser may be a readily available Internet web browser software product, such as, for example, a browser available from Netscape, Internet Explorer, or a Java-implemented browser. In an alternate embodiment, the browser may be implemented as a stand alone Java applet.

[0051] In one embodiment, the viewer uses the standard HTML language to insert an image into web pages. One embodiment of such a viewer is described in U.S. Patent Application Serial No. 09/757,561 filed January 9,

2001, entitled "Multi-Image Viewer," which is incorporated herein by reference.

[0052] In one embodiment, the viewer is implemented as a client-server system. The server stores images. The images may be stored in a compressed format. In one embodiment, the images are compressed according to a block-based integer wavelet transform entropy coding scheme. For more information on one embodiment of such a transform, see U.S. Patent No. 5,909,518, entitled "System and Method for Performing Wavelet-Like and Inverse Wavelet-Like Transformation of Digital Data," issued June 1, 1999. One embodiment of a block-based transform is described in U.S. Application Serial No. 09/358,876, entitled "Memory Saving Wavelet-Like Image Transform System and Method for Digital Camera and Other Memory Conservative Applications," filed July 22, 1999. One embodiment of scalable coding is described in U.S. Patent Application No. 5,949,911, entitled "System and Method for Scalable Coding of Sparse Data Sets," issued September 7, 1999. One embodiment of block based coding technique is described in U.S. Patent No. 5,886,651, entitled "System and Method for Nested Split Coding of Sparse Data Sets," issued March 23,

1999. Each of these are assigned to the corporate assignee of the present invention and incorporated herein by reference.

[0053] The compressed images are stored in a file structure. In one embodiment, the file structure comprises of a series of sub-images, each one being a predetermined portion of the size of its predecessor (e.g., 1/16 of the size of its predecessor). In one embodiment, each sub-picture is made up of a series of blocks that each contains the data associated with a 64x64 pixel block. That is, each image is divided into smaller individual blocks that are 64x64 pixels. Each block contains data for decoding the 64x64 block and information that can be used for extracting the data for a smaller 32x32 block. Accordingly, each sub-image contains two separate resolutions. When the image is compressed, the bit-stream is organized around these 64x64 blocks and server software extracts a variety of resolution and/or quality levels from each of these blocks.

[0054] One embodiment of a file structure along with multiresolution compressed image management that may be used with the present invention is described in U.S. Patent Application Serial No. 09/060,398, entitled "Multiresolution Compressed Image Management System and Method,"

issued March 21, 2000, assigned to the corporate assignee of the present invention and incorporated herein by reference.

[0055] The viewer allows a user to interact with an image in order to obtain versions of the image at different zoom-in levels. In one embodiment, when the user zooms in the images, the viewer calculates the new geometric coordinates for the new view. Based on the location of the cursor, the viewer calculates which part of which images will appear in the window and then obtains the appropriate data. Based on this determination, the client makes a simple request to the server and the server responds with the appropriate block(s) of data. Using the data, the viewer determines which part of each image is to appear.

[0056] In one embodiment, the viewer keeps track of which data it already has so that it does not have to request the same data multiple times from the server. In one embodiment, the viewer keeps track of what is in the window and also what other data is in the cache. There is a pixel to pixel mapping between the image and the window, so depending on resolution level, window size, and image position within (or without) the window, the client performs the geometric calculations.

[0057] In one embodiment, each image has a hypertext link so the user can click on a specific image and cause the browser to go to a new location in the image.

[0058] In one embodiment, the request for data is performed using a HTTP 'GET' command that specifies the URL of each image, which resolution level, and which blocks of data are required (based on, for example, resolution level).

[0059] In one embodiment, all data received from the server is cached locally and reused wherever possible. Caching data locally facilitates random access to different parts of the image and allows images, or parts of images, to be loaded in a variety of resolution and quality levels. In an alternative embodiment, the data is not be cached locally.

[0060] In one embodiment, the system reuses the existing image data together with the new image data to create a high quality higher resolution view. Thus, the viewer uses a file hierarchy that allows for two resolution levels to be extracted from one sub-image.

[0061] In an alternative embodiment, all the images are initially downloaded and decoded by the client. Then only that portion of each image that is to appear in the window is scaled.

[062] An Exemplary Data Management System

[063] One embodiment of a data management system that may be used to implement the techniques described herein is described in U.S. Patent Application Serial No. 09/687,467, entitled "Multi-resolution Image Data Management System and Method Based on Tiled Wavelet-Like Transform and Sparse Data Coding," filed October 12, 2000, assigned to the corporate assignee of the present invention and incorporated herein by reference.

[064] In the following description, the terms "wavelet" and "wavelet-like" are used interchangeably. Wavelet like transforms generally have spatial frequency characteristics similar to those of conventional wavelet transforms and are losslessly reversible, but have shorter filters that are more computationally efficient.

[065] The present invention may be implemented in a variety of devices that process images, including a variety of computer systems, ranging from high end workstations and servers to low end client computers as well as in application specific dedicated devices, such as digital cameras.

[066] *System for Encoding and Distributing Multi-Resolution Images*

[067] Figure 4 shows a distributed computer system, including a web server 140 and a number of client computers 120 for distributing, multi-resolution images 190 to the client computers via a global communications network 110, such as the Internet, or any other appropriate communications network, such as a local area network or Intranet. An imaging encoding workstation 150 prepares multi-resolution image files for distribution by the web server. In some embodiments, the web server 140 may also perform the image encoding tasks of the image encoding workstation 150.

[068] A typical client device 120 will be a personal digital assistant, personal computer workstation, or a computer controlled device dedicated to a particular task. The client device 120 will preferably include a central processing unit 122, memory 124 (including high speed random access memory and non-volatile memory such as disk storage) and a network interface or other communications interface 128 for connecting the client device to the web server via the communications network 110. The memory 124, will typically store an operating system 132, a browser application or other image viewing application 134, an image decoder module 180, and multi-resolution image files 190 encoded in accordance with the present

invention. In one embodiment, the browser application 134 includes or is coupled to a Java™ (trademark of Sun Microsystems, Inc.) virtual machine for executing Java language programs, and the image decoder module is implemented as a Java™ applet that is dynamically downloaded to the client device along with the image files 190, thereby enabling, the browser to decode the image tiles for viewing.

[069] The web server 140 will preferably include a central processing unit 142, memory 144 (including high speed random access memory, and non-volatile memory such as disk storage), and a network interface or other communications interface 148 for connecting the web server to client devices and to the image encoding workstation 150 via the communications network 110. The memory 141 will typically store an http server module 146 for responding to http requests, including request for multi-resolution image files 190.

[070] The web server 140 may optionally include an image processing module 168 with encoding procedures 172 for encoding images as multi-resolution images.

[071] *Computer System*

[072] Referring to Figure 5, the image processing workstation 150 may be implemented using a programmed general-purpose computer system. Figure 5 may also represent the web server, when the web server performs image processing tasks. The computer system 150 may include:

[073] one or more data processing units (CPU's) 152;

[074] memory 154 which will typically include both high speed random access memory, as well as non-volatile memory;

[075] user interface 156 including a display device 157 such as a CRT or LCD type display:

[076] a network or other communication interface 158 for communicating with other computers as well as other devices;

[077] data port 160, such as for sending and receiving images to and from a digital camera (although such image transfers might also be accomplished via the network interface 158); and

[078] one or more communication buses 161 for interconnecting the CPU(s) 152, memory 154, user interface 156, network interface 158 and data port 160.

[079] The computer system's memory 154 stores procedures and data, typically including:

[080] an operating system 162 for providing basic system services;

[081] a file system 164, which may be part of the operating system;

[082] application programs 166, such as user level programs for viewing and manipulating images.

[083] an image processing module 168 for performing various image processing functions including those that are described herein;

[084] image files 190 representing various images; and

[085] temporary image data arrays 192 for intermediate results generated during image processing and image regeneration.

[086] The computer 150 may also include a http server module 146 (Figure 4) when this computer 150 is used both for image processing and distribution of multi-resolution images. The image processing module 168 may include an image encoder module 170 and an image decoder module 180. The image encoder module 170 produces multi-resolution image files 190, the details of which will be discussed below. The image encoder module 170 may include:

[087] an encoder control program 172 which controls the process of compressing and encoding an image (starting with a raw image array 189,

which in turn may be derived from the decoding of an image in another image file format),

[088] a set of wavelet-like transform procedures 174 for applying wavelet-like filters to image data representing an image;

[089] a block classifier procedure 176 for determining the quantization divisors to be applied to each block (or band) of transform coefficients for an image;

[090] a quantizer procedure 178 for quantizing the transform coefficients for an image; and

[091] a sparse data encoding procedure 179, also known as an entropy encoding procedure, for encoding the quantized transform coefficients generated by the quantizer procedure 178.

[092] The procedures in the image processing module 168 store partially transformed images and other temporary data in a set of temporary data arrays 192.

[093] The image decoder module 180 may include:

[094] a decoder control program 182 for controlling the process of decoding an image file (or portions of the image file) and regenerating the image represented by the data in the image file;

[095] a sparse data decoding procedure 184 for decoding the encoded, quantized transform coefficients stored in an image file into a corresponding array of quantized transform coefficients;

[096] a de-quantizer procedure 186 for dequantizing a set of transform coefficients representing a tile of an image; and

[097] a set of wavelet-like inverse transform procedures 188 for applying wavelet-like inverse filters to a set of dequantized transform coefficients, representing a tile of an image, so as to regenerate that tile of the image.

[098] *Overview of Image Capture and Processing*

[099] Referring to Figure 6, raw image data 200 obtained from a digital camera's image capture mechanism (Figure 17) or from an image scanner or other device, is processed by "tiling the image data." More specifically, the raw image is treated as an array of tiles 202, each tile having a predefined size such as 64 x 64 (i.e., 64 rows by 64 columns). In other embodiments, other tile sizes, such as 32 x 32 or 16 x 32 or 128 x 128 or 64 x 128 may be used. The tiles are non-overlapping portions of the image data. A sufficient number of tiles are used to cover the entire raw image that is to be processed, even if some of the tiles overhang the edges of the raw image.

The overhanging portions of the tiles are filled with copies of boundary data values during the wavelet transform process, or alternately are filled with null data. Tile positions are specified with respect to an origin at the upper left corner of the image, with the first coordinate indicating the Y position of the tile (or a pixel or coefficient within the tile) and the second coordinate indicating the X position of the tile (or a pixel or coefficient within the tile). Thus, a tile at position 0,128 is located at the top of the image and has its origin at the 128th pixel of the top row of pixels.

[0100] A wavelet or wavelet-like decomposition transform is successively applied to each tile of the image to convert the raw image data in the tile into a set of transform coefficients. When the wavelet-like decomposition transform is a one dimensional transform that is being applied to a two dimensional array of image data, the transform is applied to the image data first in one direction (e.g., the horizontal direction) to produce an intermediate set of coefficients, and then the transform is applied in the other direction (e.g., the vertical direction) to the intermediate set of coefficients so as to produce a final set of coefficients. The final set of coefficients are the result of applying the wavelet-like decomposition transform to the image data in both the horizontal and vertical dimensions.

[0101] The tiles are processed in a predetermined raster scan order. For example, the tiles in a top row are processed going from one end (e.g., the left end) to the opposite end (e.g., the right end), before processing the next row of tiles immediately below it, and continuing until the bottom row of tiles of the raw image data has been processed.

[0102] The transform coefficients for each tile are generated by successive applications of a wavelet-like decomposition transform. A first application of the wavelet decomposition transform to an initial two dimensional array of raw image data generates four sets of coefficients, labeled LL, HL1, LH1 and HH1. Each succeeding application of the wavelet decomposition transform is applied only to the LL set of coefficients generated by the previous wavelet transformation step and generates four new sets of coefficients, labeled LL, HLx, LHx and HHx, where x represents the wavelet transform "layer" or iteration. After the last wavelet decomposition transform iteration only one LL set remains. The total number of coefficients generated is equal to the number of data samples in the original data array. The different sets of coefficients generated by each transform iteration are sometimes called layers. The number of wavelet transform layers generated for an image is typically a function of the resolution of the initial image. For

tiles of size 64 x 64, or 32 x 32, performing five wavelet transformation layers is typical, producing 16 spatial frequency subbands of data:

[0103] LL_5, HL_5, LH_5, HH_5, HL_4, LH_4, HH_4, HL_3, LH_3, HH_3, HL_2, LH_2, HH_2, HL_1, LH_1, HH_1.

[0104] The number of transform layers may vary from one implementation to another, depending on both the size of the tiles used and the amount of computational resources available. For larger tiles, additional transform layers would likely be used, thereby creating additional subbands of data. Performing more transform layers will often produce better data compression, at the cost of additional computation time, but may also produce additional tile edge artifacts.

[0105] The spatial frequency subbands are grouped as follows. Subband group 0 corresponds to the LL_N subband, where N is the number of transform layers applied to the image (or image tile). Each other subband group i contains three subbands, LH_i, HL_i, and HH_i As will be described in detail below, when the transform coefficients for a tile are encoded, the coefficients from each group of subbands are encoded separately from the coefficients of the other groups of subband. In one embodiment, a pair of bitstreams is generated to represent the coefficients in each group of

subbands. One of the bitstreams represents the most significant bit planes of the coefficients in the group of subbands while the second bitstream represents the remaining, least significant bit planes of the coefficients for the group of subbands.

[0106] The wavelet coefficients produced by application of the wavelet-like transform are preferably quantized (by quantizer 178) by dividing the coefficients in each subband of the transformed tile by a respective quantization value (also called the quantization divisor). In one embodiment, a separate quantization divisor is assigned to each subband. More particularly, as will be discussed in more detail below, a block classifier 176 generates one or more values representative of the density of features in each tile of the image, and based on those one or more values, a table of quantization divisors is selected for quantizing the coefficients in the various subbands of the tile.

[0107] The quantized coefficients produced by the quantizer 178 are encoded by a sparse data encoder 179 to produce a set of encoded subimage subfiles 210 for each tile of the image.

[0108] Details of the wavelet-like transforms used in one embodiment are below. Circuitry for performing the wavelet-like transform of the one

embodiment is very similar to the wavelet transform and data quantization methods described in U.S. Patent No. 5,909,518 entitled "System and Method for Performing Wavelet and Inverse Wavelet Like Transformations of Digital Data Using Only Add and Bit Shift Arithmetic Operations," which is hereby incorporated by reference as background information.

[0109] The sparse data encoding method of the preferred embodiment is called Nested Quadratic Splitting (NQS) and is described in detail below. This sparse data encoding method is an unproved version of the NQS sparse data encoding method described in U.S. Patent No. 5,949,911, entitled "System and Method for Scalable Coding of Sparse Data Sets," which is hereby incorporated by reference as background information.

[0110] Figure 6B depicts a mapping of spatial frequency subbands to NQS subbands used for encoding transform coefficients. In particular, in one embodiment, seven spatial frequency subbands (LL_5, HL_5, LH_5, HH_5, HL_4, LH_4, and HH_4) are mapped to a single NQS subband (subband 0) for purposes of encoding the coefficients in these subbands. In other words, the coefficients in these seven spatial frequency subbands are treated as a single top level block for purposes of NQS encoding. In one embodiment, NQS subbands 0, 1, 2 and 3 are encoded as four top level NQS blocks, the most

significant bit planes of which are stored in a bitstream representing a lowest resolution level of the image in question.

[0111] *Image Resolution Levels and Subimages*

[0112] Referring to Figure 7, an image is stored at a number of resolution levels 0 to N, typically with each resolution level differing from its neighbors by a resolution factor of four. In other words, if the highest resolution representation (at resolution level N) of the image contains X amount of information, the second highest resolution level representation N-1 contains X/4 amount of information, the third highest resolution level representation contains X/16 amount of information, and so on. The number of resolution levels stored in an image file will depend on the size of the highest resolution representation of the image and the minimum acceptable resolution for the thumbnail image at the lowest resolution level. For instance, if the full or highest resolution image is a high definition picture having about 16 million pixels (e.g., a 4096 x 4096 pixel image), it might be appropriate to have seven resolution levels: 4096 x 4096, 2048 x 2048, 1024 x 1024, 512 x 512, 256 x 256, 128 x 128, and 64 x 64.

[0113] However, as shown in Figure 4, one feature or aspect of the present invention is that when a inulti-resolution image has more than, say, three or four resolution levels, the image is encoded and stored in multiple "base image" files, each of which contains the data for two to four of the resolution levels. Alternately, all the base images may be stored in a single file, with each base image being stored in a distinct base image subfile or subfile data structure within the image file.

[0114] Each base image file (or subfle) contains the data for reconstructing a "base image" and one to three subimages (lower resolution levels). For instance, in the example shown in Figure 7, the image is stored in three tiles, with a first tile storing the image at three resolution levels, including the highest definition level and two lower levels, a second file stores the image at three more resolution levels (the fourth, fifth and sixth highest resolution levels) and a third file stores the image at the two lowest resolution levels, for a total of eight resolution levels. Generally, each successive file will be smaller than the next larger file by a factor of about 2^{2X}, where X is the number of resolution levels in the larger file. For instance, if the first file has three resolution levels, the next file will typically be smaller by a factor of $64(2^6)$.

[0115] As a result, an image file representing a group of lower resolution levels will be much smaller, and thus much faster to transmit to a client computer, than the image file containing the full resolution image data. For instance, a user of a client computer might initially review a set of thumbnail images, at a lowest resolution level (e.g., 32 x 32 or 64 x 64), requiring the client computer to review only the smallest of the three image files, which will typically contain about 0.024% as much data as the highest resolution image file. When the user requests to see the image at a higher resolution, the client computer may receive the second, somewhat larger image file, containing about 64 times as much data as the lowest resolution image file. This second file may contain three resolution levels (e.g., 512 x 512, 256 x 256, and 128 x 128), which may be sufficient for the user's needs. In the event the user needs even higher resolution levels, the highest resolution file will be sent. Depending on the context in which the system is used, the vendor of the images may charge additional fees for downloading each successively higher resolution image file.

[0116] It should be noted that many image files are not square, but rather are rectangular, and that the square image sizes used in the above examples are not intended to in any way to limit the scope of the invention. While the

basic unit of information that is processed by the image processing modules is a tile, which is typically a 64 x 64 or 32 x 32 array of pixels, any particular image may include an arbitrarily sized array of such tiles. Furthermore, the image need not be an even multiple of the tile size, since the edge tiles can be truncated wherever appropriate.

[0117] The designation of a particular resolution level of an image as the "thumbnail" image may depend on the client device to which the image is being sent. For instance, the thumbnail sent to a personal digital assistant or mobile telephone, which have very small displays, may be much smaller than (for example, one sixteenth the size of) the thumbnail that is sent to a personal computer and the thumbnail sent to a device having a large, high definition screen may be much larger than the thumbnail sent to a personal computer having a display of ordinary size and definition. When an image is to be potentially used with a variety of client devices, additional base images are generated for the image so that each type of device can initially receive an appropriately sized thumbnail image.

[0118] When an image is first requested by a client device, the client device may specify its window size in its request for a thumbnail image or the server may determine the size of the client device's viewing window by

querying the client device prior to downloading the thumbnail image data to the client device. As a result, each client device receives a minimum resolution thumbnail that is appropriately sized for that device.

[0119] *Image File Data Structures*

[0120] Referring to Figures 8A through 8E, when all the tiles of an image have been transformed, compressed and encoded, the resulting encoded image data is stored as an image file 190. The image file 190 includes header data 194 and a sequence of base image data structures, sometimes called base image subfiles 196. Each base image subfile 196 typically includes the data for displaying the image at two or more resolution levels. Furthermore, each base image supports a distinct range of resolution levels. The multiple base images and their respective subimages together provide a full range of resolution levels for the image, as conceptually represented in Figure 4. While the resolution levels supported by the base image levels are non-overlapping in one embodiment, in an alternate embodiment the resolution levels supported by one base image may overlap with tile resolution levels supported by another base image (for the same initial full resolution image).

[0121] In one embodiment, each image file 190 is an html file or similarly formatted web page that contains a link 198, such as an object tag or applet tag, to an applet 199 (e.g., a Java™ applet) that is automatically invoked when the file is downloaded to a client computer. The header 194 and a selected one of the base images 196 are used as data input to the embedded applet 199, which decodes and renders the image on the display of a user's personal digital assistant or computer. The operation of the applet is transparent to the user, who simply sees the image rendered on his/her computer display. Alternately, the applet may present the user with a menu of options including the resolution levels available with the base image subfile or subfiles included in the image file, additional base image subfiles that may be available from the server, as well as other options such as image cropping options.

[0122] In an alternate embodiment, the client workstations include an application, such as a browser plug-in application, for decoding and rendering images in the file format of the present invention. Further, each image file 210 has an associated data type that corresponds to the plug-in application. The image file 210 is downloaded along with an html or similarly formatted web page that includes an embed tag or object tag that

points to the image file. As a result, when the web page is downloaded to a client workstation, the plug-in application is automatically invoked and executed by the client computer's. As a result, the image file is decoded and rendered and the operation of the plug-in application is transparent to the user.

[0123] The image file 190-A shown in Figure 8A represents one possible way of storing a multi-resolution image, and is particularly suitable for storing a multi-resolution image in a server. In a client computer, the image file 190-B as shown in Figure 8B may contain only one base image 196. In addition, the client version of the image file 190 may contain a link 201 to the image file 190-A in the server. The link 201 is used to enable a user of the client computer to download other base images (at other resolution levels) of the same image. Alternately, the link 201 is a Java™ (trademark of Sun Microsystems) script for requesting an image file containing any of the higher resolution base images from the web server. If there is a charge for obtaining the higher resolution image file, the script will invoke the execution of the server procedure for obtaining payment from the requesting user.

[0124] In yet another alternate embodiment, a multi-resolution image may be stored in the server as a set of separate base image tiles 190-B, each having the format shown in Figure 8B. This has the advantage of providing image tiles l90-B that are ready for downloading to client computers without modification.

[0125] Referring to Figure 8A again, the header 194 of the image tile includes the information needed to access the various base image subfiles 196. In particular, in one embodiment, the header 194 stores:

[0126] an identifier or the URL of the image file in the server;

[0127] a parameter value that indicates the number of base image subfiles 196 in the file (or the number of base image files in embodiments in which each base image is stored in a separate file);

[0128] the size of each base image data structure; and

[0129] a offset pointer to each base image data structure (or a pointer to each base image file in embodiments in which each base image is stored in a separate file).

[0130] Each base image subfile 196 has a header 204 and a sequence of bitstreams 206. The bitstreams are labeled la, lb, to N, where N is the number of resolution levels supported by the base image in question. The

meaning of the labels "la" and the like will be explained below. The information in each bit stream 206 will be described in full detail below. The header data 204 of each base image subfile includes fields that indicate:

[0131] the size of the base image subfile (i.e., the amount of storage occupied by the base image subfile);

[0132] the size of the tiles (e.g., the number of rows and columns of pixels) used to tile the base image, where each tile is separately transformed and encoded, as described below;

[0133] the color channel components stored for this base image subfile;

[0134] the transform filters used to decompose the base image (e.g., different sets of transform filters may be used on different images);

[0135] the number of spacial frequency subbands encoded for the base image (i.e., for each tile of the base image);

[0136] the number of resolution levels (else called subimages) supported by the base image;

[0137] the number of bitstreams encoded for the base image (i.e., for each tile of the base image); and

[0138] information for each of the bitstreams.

[0139] The header information far each bitstream in the base image subfile may include:

[0140] an offset pointer to the bitstream to indicate its position within the image tile (or within the base image subfile);

[0141] the size of bitstream (how much data is in the bitstream);

[0142] the range of spatial frequency subbands included in the bitstream;

[0143] the number of color channels in the bitstream;

[0144] the range of bit planes included in the bitstream, which indicates how the bit planes of the coefficients in the subbands were divided between significant, insignificant and possibly mid-significant portions; and a table of offset pointers to the tiles 208 within the bitstream.

[0145] Each bitstream 206 includes a sequence of tile subarrays 208, each of which captains the i^{th} bitstream for a respective tile of the image. The bitstream 206 may optionally include a header 209 having fields used to override parameters specified for the base image by the base image header 204. When the image file contains a cropped image, the set of tile subarrays 208 included to the image file is limited to those needed to represent the cropped image.

[0146] In one embodiment, the image file header 194 also includes parameters indicating "cropped image boundaries." This is useful for partial copies of the image file that contain data only for a cropped portion of the image, which in turn is very useful when a client computer is being used to perform pan and zoom operations in an image. For instance, a user may have requested only a very small portion of the overall image, but at very high resolution. In this case, only the tiles of the image needed to display the cropped portion of the image will be included in the version of the image tile sent to the user's client computer, and the cropped image boundary parameters are used to convey this information to the procedures that render the image an the client computer. Two types of image cropping information are provided by the image file header 194: cropping that applies to the entire image file, and any further cropping that applies to specific subimages. For instance, when a client computer first receives an image, it may receive just the lowest resolution level subimage of a particular base image, and that subimage will typically not be cropped (compared to the full image). When the client zooms in on a part of the image at a specified higher resolution level, only the tiles of data needed to generate the portion of the image to be viewed on the client computer are

sent to the client computer, and thus new cropping parameters will be added to the header of the image file stored (or cached) in the client computer to indicate the cropping boundaries for the subimage level or levels downloaded to the client computer in response to the client's image zoom command.

[0147] The table of offset pointers to tiles that is included in the base image header for each bitstream in the base image is also used during zooming and panning. In particular, referring to Figure 18, when an image file is first downloaded by a client computer or device (240), the higher level bitstreams may be unpopulated, and thus the table of offset pointers will initially contain null values. When the user of the client devices zooms in on the image, the data for various tiles of the higher level bitstreams are downloaded to the client device, as needed (242), and the table of offset pointers to tiles is updated to reflect the tiles for which data have been downloaded to the client computer. When the client further pans across the image at the zoomed or higher resolution level, additional tiles of information are sent to the client computer as needed, and the cropping information in the image tile header l94 and the tile offset information in the

base image header are again updated to reflect the tiles of data stored for each bitstream (244).

[0148] Referring again to Figures 8A-8E, the information in the headers of the image file and the base image subfiles enables quick indexing into any part of the tile, which enables a computer or other device to locate the beginning or end of any portion of the image, at any resolution level, without having to decode the contents of any other portions of the image file 190. This is useful, for example, when truncating the image file 190 so as to generate a lower image quality version of the file, or a cropped image version of the file, such as for transmission over a communications network to another computer or device.

[0149] In some of the discussions that follow, the terms "subimage" and "differential subimage" will be used with respect to the bitstreams 206 as follows. Generally, any subimage of a base image will include all the bitstreams from bitstream 1a through a particular last bitstream, such as bitstream 3. This group of contiguous bitstreams constitute the data needed to reconstruct the image at a particular resolution level, herein called a subimage. A "differential subimage" consists of the additional bitstreams needed to increase the image resolution from one subimage level to the next.

For instance, bitstreams 1c, 2b and 3 might together be called a differential subimage because these bitstreams contain the data needed to double the resolution of the subimage generated from bitstreams 1a through 2a.

[0150] Referring to Figure 8C, the encoded data 190-C representing a base image is initially stored in "tile order." The image file 190-C includes a header 222 and a set of tile subfiles 220. Referring to Figure 8D, each tile subfile 220 contains a header 224 denoting the quantization table used to encode the tile, offset pointers to the bitstreams within the subfile, and other information. The title subfile 220 for each tile also contains a set of bitstream subarrays 226. Each tile bitstream subarray 226 contains encoded data representing either the most significant bit planes, least significant bit planes or a middle set of bit planes or a respective set of NQS subbands (see Figure 6B) of the tile. The following table shows an example of bit plan mappings to bitstream subarrays:

[0151]

NQS Subbrand Nos. Resolution	0 to 3	4, 5, 6	7, 8, 9
16 x 16	S		
32 x 32	S + MS	S	
64 x 64	S + MS + IS	S + IS	All

[0152] In this table, the bit planes corresponding to S, MS and IS differ for each NQS subband. These bit plane ranges are specified in the header of the base image subfile. For instance, for NQS subbands 0 to 3, S may corresponding to bit planes 16 to 7, MS may correspond to bit planes 6 to 4, and IS may correspond to bit planes 3 to 0, while for NQS subbands 4 to 6, S may corresponding to bit planes 16 to 5, and IS may correspond to bit planes 4 to 0.

[0153] Bitstreams 1a, 1b and 1c contain the encoded data representing the most significant, middle and least significant bit planes of NQS subbands 0, 1, 2 and 3, respectively. Bitstreams 2a and 2b contain the encoded data representing the most significant and least significant bit planes, respectively, of NQS subbands 4, 5 and 6, which correspond to the LH_2, HL_2 and HH_2 subbands. Bitstream 3 contains all the bit planes of the encoded data representing NQS subbands 7, 8 and 9, which correspond to the LH_1, HL_1 and HH_1 subbands, respectively.

[0154] The tile subfiles 220 may be considered to be "temporary" files, because the encoded tile data is later reorganized from the file format of Figsures 8C and 8D into the file format shown in Figure 8A.

[0155] Figure 8E shows a specific example of a base image subfile 196, labeled 196A. The base image subfile contains twelve bitstreams 206, which are used to generate the base image and two lower resolution subimages. The base image has been transformed with five layers of wavelet transforms, providing sixteen spatial frequency subbands of data, which have been encoded and organized into three subimages, including the base image. The number of subimages is somewhat arbitrary, since the subbands generated by five transform layers could be used to generate as many as six subimages. However, using this base image subfile to generate very small subimages is not efficient in terms of memory or storage utilization, and therefore it will be preferred to use a smaller base image subfile to generate smaller subimages.

[0156] In Figure 8E, the base image has been processed by five transform layers, but the resulting data has been organized into just three subimage levels instead of six. Effectively, the last three transform layers, which convert subband LL_2 into ten subbands (LL_5, LH_5, HL_5, HH_5, LH_4, HL_4, HH_4, LH_3 and HH_3), are not used to generate an extra subimage level. Rather, the last three transform layers are used only to produce better data compression.

[0157] As shown in Figure 8E, when the five transform layers of image data are mapped to three subimages, the mapping of bitstream data subarrays 206 to subimages is as follows:

[0158] subimage 0, the lowest level subimage, corresponds to bitstream subarray 206-1a, which contains the most significant bit planes of NQS subbands 0 to 3 (see Figure 6B);

[0159] subimage 1 corresponds to bitstreams 206-1a, 206-1b and 206-2a; and

[0160] subimage 2, the base image, corresponds to all the bitstreams 206 in the base image subfile.

[0161] When the transform layers are mapped to more subimages (subimage levels) than in the example shown in Figure 8E, the first bitstream 206-1a will include fewer of the spatial frequency subbands.

[0162] A sparse data encoding technique is used to encode the transform coefficients for each group of subbands of each tile so that it takes very little data to represent arrays of data that contain mostly zero values. Typically, higher frequency portions (i.e., subbands) of the transformed, quantized image data will contain more zero values than non-zero values, and further most of the non-zero values will have relatively small absolute value.

Therefore, the higher level bit planes of many tiles will be populated with very few non-zero bit values.

[0163] *Tiled Wavelet Transform Method*

[0164] Referring to Figure 9, the process for generating an image file begins when an image is captured by the image capture device (step 250). If the image size is variable, the size of the captured image is determined and the number of rows and columns of tiles needed to cover the image data is determined (step 252). If the image size is always the same, step 252 is not needed.

[0165] Next, all the tiles in the image are processed in a predetermined order for example in raster scan order, by applying a wavelet-like decomposition transform to them in both the horizontal and vertical directions, then quantizing the resulting transform coefficients, and finally by encoding the quantized transform coefficients using a sparse data compression and encoding procedure (step 254). The encoded data for each tile is stored in a temporary file or subfile, such as in the format shown in Figure 8D.

[0166] After all the tiles in the image have been processed, a multi-resolution image file containing all the encoded tiles is stored in non-volatile memory

(step 256). More specifically, the encoded tile data from the temporary files is written into an output bitstream file in resolution reversed order, in the file format shown in Figure 8A. "Resolution reversed order" means that the image data is stored in the file with the lowest resolution bitstream first, followed by the next lowest resolution bitstream, and so on.

[0167] The wavelet-like decomposition transform used in step 254 is described in more detail below, with reference to Figures 10A, 10B and 10C. The quantization and sparse data encoding steps are also described in detail below.

[0168] After the initial image has been processed, encoded and stored as a multi-resolution image file, typically containing two to four resolution levels, if more than one base image is to be included in the image file (257), the original image is down-sampled and anti-aliased so as to generate a new base image (258) that is smaller in each dimension by a factor of 2^X, where X is the number of subimage levels in the previously generated multi-resolution image file. Thus, the new base image will be a factor of 4 smaller than the smallest lowest-resolution subimage of the base image. The new base image is then processed in the same way as the previous base image so as to generate an additional, but much smaller, encoded multi-

resolution base image that is added to the image file. If the original base image had sufficiently high resolution, a third base image may be formed by performing a second round of down-sampling and anti-aliasing, and a third encoded multi-resolution base image file may be stored in the image file. The last encoded base image may contain fewer subimage levels than the others, and in some embodiments may contain only a single resolution level, in which case that image file is effectively a thumbnail image file.

[0169] In an alternate embodiment, each encoded base image is stored in a separate image file, and these image files are linked to each other either by information stored in the headers of the image files, or by html (or html-like) links.

[0170] In one embodiment, the down-sampling filter is a one-dimensional FIR filter that is applied first to the rows of the image and then to the columns, or vice versa. For example, if the image is to be down-sampled by a factor of 4 in each dimension (for a factor of 16 reduction in resolution), the FIR filter may have the following filter coefficients:

[0171] Filter A = (-3 -4 -4 10 10 29 29 29 29 10 10 -4 -4 -3 -3) 1 /128.

[0172] This exemplary filter is applied to a set of 14 samples at a time to produce one down-sampled value, and is then shifted by four samples and

is then applied again. This repeats until L/4 down-sampled values have been generated, where L is the number of initial samples (i.e., pixel values). At the edges of the image data array, reflected data is used for the filter coefficients that extend past the edge of the image data. For instance, at the left (or top) edge of the array, the first six coefficients are applied to reflected data values, tile four "29/128", coefficients are applied to the first four pixel values in the row (or column) being filtered, and the last six coefficients are applied to the next six pixels in the row (or column).

[0173] If an image is to be down-sampled by a factor of 8, the above described filter is applied to down-sample by a factor of 4, and then a second filter is applied to further down-sample the image data by another factor of 2. This second filter, in one embodiment, is a FIR filter that has the following filter coefficients:

[0174] Filter B = (-3 -4 10 29 29 10 -4 -3) 1/64.

[0175] Alternately, a longer filter could be used to achieve the down-sampling by a factor of 8 in one filter pass.

[0176] The down-sampling filters described above have the following properties: they are low-pass filters with cut-off frequencies at one quarter and one half the Nyquist frequency, respectively; each filter coefficient is

defined by a simple fraction in which the numerator is an integer and the denominator is a positive integer power of 2 (i.e., a number of the form 2^N, where N is a positive integer). As a result of these filter properties, the down-sampling can be performed very efficiently while preserving the spatial frequency characteristics of the image and avoiding aliasing effects.

[0177] While the order in which the down-sampling filter(s) are applied to an array of image data (i.e., rows and then columns, or vice versa) will affect the specific down-sampled pixel values generated, the effect on the pixel values is not significant. Other down-sampling filters may be used in alternate embodiments.

[0178] *Wavelet-Like Decomposition Using Edge, Interior and Center Transform Filters*

[0179] Figure 10A-10C schematically represent the process of performing a wavelet-like decomposition on a set of image data X_0 to X_{2n-1} to generate a set of coefficients L_0 to L_{n-1} and H_0 to H_{n-1} where the L coefficients represent the low spatial frequency components of the image data and the H coefficients represent the high spatial frequency components of the image data.

[0180] In one embodiment, the wavelet-like transform that is applied is actually two filters. A first filter, T1, called the edge filter, is used to generate the first two and last two coefficients in the row or column of transform coefficients that are being generated, and a second filter T2, called the interior filter, is used to generate all the other coefficients in the row or column of transform coefficients being generated. The edge filter, T1 is a short filter that is used to transform data at the edges of a tile or block, while the interior filter T2 is a longer filter that is used to transform the data away from the edges of the tile or block. Neither the edge filter nor the interior filter uses data from outside the tile or block. As a result, the working memory required to apply the wavelet-like transform described herein to an array of image data is reduced compared to prior art systems. Similarly, the complexity of the circuitry and/or software for implementing the wavelet-like transform described herein is reduced compared to prior art systems.

[0181] In one embodiment, the edge filter includes a first, very short filter (whose "support" covers two to four data values) for generating the first and last coefficients, and a second filter for generating the second and second to last coefficients. The second edge filter has a filter support that extends over

three to six data values, and thus is somewhat longer than the first edge filter but shorter than the interior filter T2. The interior filter for generating the other coeffcients typically has a filter support of seven or more data values. The edge filter, especially the first edge filter for generating the first and last high spatial frequency coefficient values, is designed to reduce, or possibly even minimize, edge artifacts while not using any data from neighboring tiles or blocks, at a cost of decreased data compression. Stated in another way, the edge filter of the present invention is designed to ensure accurate reproduction of the edge values of the data array being processed, which in turn reduces, and possibly minimizes, edge artifacts when the image represented by the data array is regenerated.

[0182] In one embodiment, the wavelet-like decomposition transform applied to a data array includes a layer 1 wavelet-like transform that is distinct from the wavelet-like transform used when performing layers 2 to N of the transform. In particular, the layer 1 wavelet-like transform uses shorter filters, having shorter filter supports, than the filters used for layers 2 to N. One of the reasons for using a different wavelet-like transform (i.e., a set of transform filters) for layer 1 than for the other layers is to reduce or minimize rounding errors introduced by the addition of a large number of

scaled values. Rounding errors, which occur primarily when filtering the raw image data during the layer 1 transform can sometimes cause noticeable degradation in the quality of the image regenerated from the encoded image data.

[0183] The equations for the wavelet-like decomposition transform used in the preferred embodiment are presented below.

[0184] Layer 1 Forward Wavelet-Like Transform

[0185] *T1 and T2 Forward Transforms (Low Frequency):*

[0186] $Y_k = X_{2k} - X_{2k+1}$ k=0,1, …, n-1

[0187] $L_k = X_{2k+1} + \left[\frac{Y_k + 1}{2}\right] = \frac{X_{2k} + X_{2k+1} + 1}{2}$ k=0,1, …, n-1

[0188] *T1 Forward Transform (Edge Filter - High Frequency):*

[0189] $H_0 = Y_0 + \left[\frac{-L_0 + L_1 + 1}{2}\right]$

[0190] $H_1 = Y_1 + \left[\frac{-L_0 + L_2 + 2}{4}\right]$

[0191] $H_{n-2} = Y_{n-2} + \left[\frac{-L_{n-3} + L_{n-1} + 2}{4}\right]$

[0192] $$H_{n-1} = Y_{n-1} + \left[\frac{-L_{n-2} + L_{n-1} + 1}{2}\right]$$

[0193] *T2 Forward Transform (Interior Filter - High Frequency):*

[0194] $$H_k = Y_k + \left[\frac{3L_{k-2} - 22L_{k-1} + 22L_{k+1} - 3L_{k+2} + 32}{64}\right] \qquad k=2, \ldots, n-3$$

[0195] Layer 1 Inverse Wavelet-Like Transform

[0196] *T1 Inverse Transform (Edge Filter - High Frequency):*

[0197] $$Y_0 = H_0 - \left[\frac{-L_0 + L_1 + 1}{2}\right]$$

[0198] $$Y_1 = H_1 - \left[\frac{-L_0 + L_2 + 2}{4}\right]$$

[0199] $$Y_{n-2} = H_{n-2} - \left[\frac{-L_{n-3} + L_{n-1} + 2}{4}\right]$$

[0200] $$Y_{n-1} = H_{n-1} - \left[\frac{-L_{n-2} + L_{n-1} + 1}{2}\right]$$

[0201] *T2 Inverse Transform (Interior Filter):*

[0202] $$Y_k = H_k - \left[\frac{3L_{k-2} - 22L_{k-1} + 22L_{k+1} - 3L_{k+2} + 32}{64}\right] \qquad k=2, \ldots, n-3$$

[0203] $X_{2k+1} = L_k - \left[\frac{Y_k + 1}{2}\right]$ k=0,1, …, n-1

[0204] $X_{2k} = Y_k + X_{2k+1}$ k=0,1, …, n-1

[0205] Forward Wavelet-Like Transform: Layers 2 to N

[0206] The equations for one embodiment of the forward wavelet-like decomposition transform for transform levels 2 through N (i.e., all except level 1) are shown next. Note that "2n" denotes the width of the data, as measured in data samples, that is being processed by the transform; "n" is assumed to be a positive integer. The edge filter T1 is represented by the equations for H_0, H_{n-1}, L_0, and L_{n-1}, and has a shorter filter support than the interior filter T2.

[0207] In alternative embodiment, the same wavelet-like decomposition transforms are used for all layers. For example, the wavelet-like decomposition transform filters shown here are layers 2 to N would also be used for the layer 1 decomposition (i.e., for filtering the raw image data).

[0208] $H_0 = X_1 - \left[\frac{X_0 + X_2 + 1}{2}\right]$ (edge filter)

[0209] $$H_k = X_{2k+1} - \left[\frac{9\left(X_{2k} + X_{2k+2}\right) - X_{2k-2} - X_{2k+4} + 8}{16}\right] \qquad k=1, \ldots, \frac{n}{2}-3$$

[0210] $$H_{\frac{n}{2}-2} = X_{n-3} - \left[\frac{X_{n-4} + X_{n-2} + 1}{2}\right]$$ (center filter)

[0211] $$H_{\frac{n}{2}-1} = X_{n-1} - \left[\frac{11X_{n-2} + 5X_{n+1} + 8}{16}\right]$$ (center filter)

[0212] $$H_{\frac{n}{2}} = X_n - \left[\frac{5X_{n-2} + 11X_{n+1} + 8}{16}\right]$$ (center filter)

[0213] $$H_{\frac{n}{2}+1} = X_{n+2} - \left[\frac{X_{n+1} + X_{n+3} + 1}{2}\right]$$ (center filter)

[0214] $$H_k = X_{2k} - \left[\frac{9\left(X_{2k-1} + X_{2k+1}\right) - X_{2k-3} - X_{2k+3} + 8}{16}\right] \quad k = \frac{n}{2} + 2, \ldots, \text{n-2}$$

[0215] $$H_{n-1} = X_{2n-2} - \left[\frac{X_{2n-3} + X_{2n-1} + 1}{2}\right]$$ (edge filter)

[0216] $$L_0 = X_0 + \left[\frac{H_0 + 2}{4}\right] = \frac{7X_0 + 2X_1 - X_2 + 3}{8}$$ (edge filter)

[0217] $$L_1 = X_2 + \left[\frac{H_0 + H_1 + 2}{4}\right]$$ (edge filter)

[0218] $$L_k = X_{2k} + \left[\frac{5(H_{k-1} + H_k) - H_{k-2} - H_{k+1} + 8}{16}\right]$$ k=1, ..., $\frac{n}{2} - 3$

[0219] $$L_{\frac{n}{2}-2} = X_{n-4} + \left[\frac{H_{\frac{n}{2}-3} + H_{\frac{n}{2}-2} + 2}{4}\right]$$ (center filter)

[0220] $$L_{\frac{n}{2}-1} = X_{n-2} + \left[\frac{2H_{\frac{n}{2}-2} + 2H_{\frac{n}{2}-1} - H_{\frac{n}{2}} + 4}{8}\right]$$ (center filter)

[0221] $$L_{\frac{n}{2}} = X_{n-1} + \left[\frac{2H_{\frac{n}{2}+1} + 2H_{\frac{n}{2}} - H_{\frac{n}{2}-1} + 4}{8}\right]$$ (center filter)

[0222] $$L_{\frac{n}{2}+1} = X_{n+3} + \left[\frac{H_{\frac{n}{2}+1} + H_{\frac{n}{2}+2} + 2}{4}\right]$$ (center filter)

[0223] $$L_k = X_{2k+1} + \left[\frac{5(H_k + H_{k+1}) - H_{k-1} - H_{k+2} + 8}{16}\right] \qquad k = \frac{n}{2} + 2, \ldots, \text{n-3}$$

[0224] $$L_{n-2} = X_{2n-3} + \left[\frac{H_{n-2} + H_{n-1} + 2}{4}\right]$$ (edge filter)

[0225] $$L_{n-1} = X_{2n-1} + \left[\frac{H_{n-1} + 2}{4}\right] = \frac{7X_{2n-1} + 2X_{2n-2} - X_{2n-3} + 3}{8}$$ (edge filter)

[0226] The general form of the decomposition transform equations, shown above, applies only when n is at least ten. When n is less than ten, some of the equations for terms between the edge and middle terms are dropped because the number of coefficients to be generated is too few to require use of those equations. For instance, when n=8, the two equations for generating L_k will be skipped.

[0227] *Discussion of Attributes of Transform Filter*

[0228] It is noted that the edge transform filter T1 for generating L_0 and L_{n-1} has a filter support of just three input samples at the edge of the input data array, and is weighted so that 70% of the value of these coefficients is attributable to the edge value X_0 and X_{2n-1} at the very boundary of the aray of data being filtered. The heavy weighting of the edge input datum (i.e., the sample closest to the array boundary) enables the image to be reconstructed from the transform coefficients substantially without the boundary artifacts, despite the fact that the edge and interior filters are applied only to data within the tile when generating the transform coefficients for the tile. The layer 1 edge transform filter T1 for generating L_0 and L_{n-1} is weighted so that 50% of the value of these coefficients is attributable to the edge value X_{2n-1} at the very boundary of the data array being filtered.

[0229] The interior transform filters in one embodiment are not applied in a uniform manner across the interior of the data array being filtered. Furthermore, the interior filter includes a center filter for generating four high pass and four low pass coefficients at or near the center of the data array being filtered. In alternative embodiments, the center filter may generate as few as two high pass and two low pass coefficients. The center filter is used to transition between the left and right (or upper and lower)

portions of the interior filter. The transition between the two forms of the interior filter is herein called "filter switching." One half of the interior filter, excluding the center filter, is centered on even numbered data or coefficient positions while the other half of the interior filter is centered on data at odd data positions. (The even and odd data positions of the array are, of course, alternating data positions.) While the equations as written place the center filter at the middle of the array, the center filter can be positioned anywhere within the interior of the data array, so long as there is a smooth transition between the edge filter and the interior filter. Of course, the inverse transform filter must be defined so as to have an inverse center filter at the same position as the forward transform filter.

[0230] *Transform Equations for Small data Arrays, for Layers 2 to N*

[0231] When n is equal to four, the transform to be performed can be represented as:

[0232] $(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) \Rightarrow (L_0, L_1, L_2, L_3; H_0, H_1, H_2, H_3)$

[0233] and the above general set of transform equations is reduced to the following:

[0234] $H_0 = X_1 - \left[\frac{X_0 + X_2 + 1}{2}\right]$

[0235] $$H_1 = X_3 - \left[\frac{11X_2 + 5X_5 + 8}{16}\right]$$

[0236] $$H_2 = X_4 - \left[\frac{5X_2 + 11X_5 + 8}{16}\right]$$

[0237] $$H_3 = X_6 - \left[\frac{X_5 + X_7 + 1}{2}\right]$$

[0238] $$L_0 = X_0 + \left[\frac{H_0 + 2}{4}\right]$$

[0239] $$L_1 = X_2 + \left[\frac{2H_0 + 2H_1 - H_2 + 4}{8}\right]$$

[0240] $$L_2 = X_5 + \left[\frac{2H_3 + 2H_2 - H_1 + 4}{8}\right]$$

[0241] $$L_3 = X_7 + \left[\frac{H_3 + 2}{4}\right]$$

[0242] When n is equal to two, the transform can be represented as:

[0243] $(X_0, X_1, X_2, X_3) \Rightarrow (L_0, L_1; H_0, H_1)$

[0244] and the above general set of transform equations is reduced to the following:

[0245]

$$H_0 = X_1 - \left[\frac{X_0 + X_3 + 1}{2}\right]$$

$$H_1 = X_2 - \left[\frac{X_0 + X_3 + 1}{2}\right]$$

$$L_0 = X_0 + \left[\frac{H_0 + 2}{4}\right]$$

$$L_1 = X_3 + \left[\frac{H_0 + 2}{4}\right]$$

[0246] *Inverse Wavelet-Like Transform: Layers 2 to N*

[0247] The inverse wavelet-like transform for transform layers 2 through N (i.e., all except layer 1), used in one embodiment, are shown next.

[0248] The general form of the transform equations applied only when n is at least ten. When n is less than ten, some of the equations for terms between the edge and middle terms are dropped because the number of coefficients to be generated is too few to require use of those equations.

[0249] $X_0 = L_0 - \left[\frac{H_0 + 2}{4}\right]$

[0250] $$X_2 = L_1 - \left[\frac{H_0 + H_1 + 2}{4}\right]$$

[0251] $$X_{2k} = L_k - \left[\frac{5(H_{k-1} + H_k) - H_{k-2} - H_{k-1} + 8}{16}\right] \qquad k = 2, ..., \frac{n}{2} - 3$$

[0252] $$X_{n-4} = L_{\frac{n}{2}-2} - \left[\frac{H_{\frac{n}{2}-3} + H_{\frac{n}{2}-2} + 2}{4}\right]$$

[0253] $$X_{2k-1} = L_k - \left[\frac{5(H_k + H_{k+1}) - H_{k-1} - H_{k+2} + 8}{16}\right] \qquad k = \frac{n}{2} + 2, ..., n - 3$$

[0254] $$X_{n-2} = L_{\frac{n}{2}-1} - \left[\frac{2H_{\frac{n}{2}-2} + 2H_{\frac{n}{2}-1} - H_{\frac{n}{2}} + 4}{8}\right]$$

[0255] $$X_{n+1} = L_{\frac{n}{2}} - \left[\frac{2H_{\frac{n}{2}+1} + 2H_{\frac{n}{2}} - H_{\frac{n}{2}-1} + 4}{8}\right]$$

[0256] $$X_{n+3} = L_{\frac{n}{2}+1} - \left[\frac{H_{\frac{n}{2}+1} + H_{\frac{n}{2}+2} + 2}{4}\right]$$

[0257] $$X_{2n-3} = L_{n-2} - \left[\frac{H_{n-2} + H_{n-1} + 2}{4}\right]$$

[0258] $$X_{2n-1} = L_{n-1} - \left[\frac{H_{n-1} + 2}{4}\right]$$

[0259] $$X_1 = H_0 - \left[\frac{X_0 + X_2 + 1}{2}\right]$$

[0260] $$X_{2k+1} = H_k + \left[\frac{9(X_{2k} + X_{2k+2}) - X_{2k-2} - X_{2k+4} + 8}{16}\right] \qquad k = 1, \ldots, \frac{n}{2} - 3$$

[0261] $$X_{n-3} = H_{\frac{n}{2}-2} + \left[\frac{X_{n-4} + X_{n-2} + 1}{2}\right]$$

[0262] $$X_{n-1} = H_{\frac{n}{2}-1} + \left[\frac{11X_{n-2} + 5X_{n-1} + 8}{16}\right]$$

[0263] $$X_n = H_{\frac{n}{2}} + \left[\frac{5X_{n-2} + 11X_{n-1} + 8}{16}\right]$$

[0264] $$X_{n+2} = H_{\frac{n}{2}+1} + \left[\frac{X_{n+1} + X_{n+3} + 1}{2}\right]$$

[0265] $$X_{2k} = H_k + \left[\frac{9(X_{2k-1} + X_{2k+1}) - X_{2k-3} - X_{2k+3} + 8}{16}\right] \qquad k = \frac{n}{2} + 2, \ldots, n - 2$$

[0266] $$X_{2n-2} = H_{n-1} + \left[\frac{X_{2n-3} + X_{2n-1} + 1}{2}\right]$$

[0267] When n is equal to eight, the above general set of inverse transform equations is reduced to the following:

[0268] $$X_0 = L_0 - \left[\frac{H_0 + 2}{4}\right]$$

[0269] $$X_2 = L_1 - \left[\frac{H_0 + H_1 + 2}{4}\right]$$

[0270] $$X_4 = L_2 - \left[\frac{H_1 + H_2 + 2}{4}\right]$$

[0271] $$X_6 = L_3 - \left[\frac{2H_2 + 2H_3 - H_4 + 4}{8}\right]$$

[0272] $$X_9 = L_4 - \left[\frac{2H_5 + 2H_4 - H_3 + 4}{8}\right]$$

[0273] $$X_{11} = L_5 - \left[\frac{H_5 + H_6 + 2}{4}\right]$$

[0274] $$X_{13} = L_6 - \left[\frac{H_6 + H_7 + 2}{4}\right]$$

[0275] $$X_{15} = L_7 - \left[\frac{H_7 + 2}{4}\right]$$

[0276] $$X_1 = H_0 + \left[\frac{X_0 + X_2 + 1}{2}\right]$$

[0277] $$X_3 = H_1 + \left[\frac{9(X_2 + X_4) - X_0 - X_6 + 8}{16}\right]$$

[0278] $$X_5 = H_2 + \left[\frac{X_4 + X_6 + 1}{2}\right]$$

[0279] $$X_7 = H_3 + \left[\frac{11X_6 + 5X_9 + 8}{16}\right]$$

[0280] $$X_8 = H_4 + \left[\frac{5X_6 + 11X_9 + 8}{16}\right]$$

[0281] $$X_{10} = H_5 + \left[\frac{X_9 + X_{11} + 1}{2}\right]$$

[0282] $$X_{12} = H_6 + \left[\frac{9(X_{11} + X_{13}) - X_9 - X_{15} + 8}{16}\right]$$

[0283] $$X_{14} = H_7 + \left[\frac{X_{13} + X_{15} + 1}{2}\right]$$

[0284] When n is equal to four, the inverse transform to be performed can be represented as:

[0285] $$(L_0, L_1, L_2, L_3; H_0, H_1, H_2, H_3) \Rightarrow (X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7)$$

[0286] and the above general set of inverse transform equations is reduced to the following:

[0287] $$X_0 = L_0 - \left[\frac{H_0 + 2}{4}\right]$$

[0288] $$X_2 = L_1 - \left[\frac{2H_0 + 2H_1 - H_2 + 4}{8}\right]$$

[0289] $$X_5 = L_2 - \left[\frac{2H_3 + 2H_2 - H_1 + 4}{8}\right]$$

[0290] $$X_7 = L_3 - \left[\frac{H_3 + 2}{4}\right]$$

[0291] $$X_1 = H_0 + \left[\frac{X_0 + X_2 + 1}{2}\right]$$

[0292] $$X_3 = H_1 + \left[\frac{11X_2 + 5X_5 + 8}{16}\right]$$

[0293] $$X_4 = H_2 + \left[\frac{5X_2 + 11X_5 + 8}{16}\right]$$

[0294] $$X_6 = H_3 + \left[\frac{X_5 + X_7 + 1}{2}\right]$$

[0295] When n is equal to two, the inverse transform to be performed can be represented as:

[0296] $$(L_0, L_1; H_0, H_1) \Rightarrow (X_0, X_1, X_2, X_3, X_4)$$

[0297] and the above general set of inverse transform equations is reduced to the following:

[0298] $$X_0 = L_0 - \left[\frac{H_0 + 2}{4}\right]$$

[0299] $$X_3 = L_1 - \left[\frac{H_1 + 2}{4}\right]$$

[0300] $$X_1 = H_0 + \left[\frac{X_0 + X_3 + 1}{2}\right]$$

[0301] $$X_2 = H_1 + \left[\frac{X_0 + X_3 + 1}{2}\right]$$

[0302] In one embodiment, during each layer of the inverse transform process the coefficients at the even positions (i.e., the X_{2i} values) must be computed before the coefficients at the odd positions (i.e., the X_{2i+1} values).

[0303] In an alternate embodiment, the short T1 decomposition transform is used to filter all data, not just the data at the edges. Using only short T1 decomposition transform reduces computation time and complexity, but decreases the data compression achieved and thus results in larger image files. Using only short transform also reduces the computation time to decode an image file that contains an image encoded using the present invention, because only the corresponding short T1 reconstruction transform is used during image reconstruction.

[0304] *Adaptive Blockwise Quantization*

[0305] Referring to Figure 6, each wavelet coefficient produced by the wavelet-like decomposition transform is quantized:

[0306] $$\hat{x}_q = \mathrm{sign}(x)\left[\left(\frac{|x|}{q} + \frac{3}{8}\right)\right]$$

[0307] where q is the quantization divisor, and is dequantized:

[0308] $$\hat{x} = q\,\hat{x}_q .$$

[0309] In one embodiment, a quantization table is used to assign each subband of the wavelet coefficients a quantization divisor, and thus controls the compression quality. If five layers of wavelet transforms are performed for luminance values (and four layers for the chrominance values), there are 16 subbands in the decomposition for the luminance values:

[0310] $LL_5, HL_5, LH_5, HH_5, HL_4, LH_4, HH_4, HL_3, LH_3, HH_3, HL_2, LH_2, HH_2, HL_1, LH_1, HH_1$

[0311] and 13 subbands for the chrominance values:

[0312] $LL_4, HL_4, LH_4, HH_4, HL_3, LH_3, HH_3, HL_2, LH_2, HH_2, HL_1, LH_1, HH_1$

[0313] One possible quantization table for luminance values is:

[0314] q=(16, 16, 16, 18, 18, 18, 24, 24, 24, 36, 46, 46, 93, 300, 300, 600)

[0315] and for the chrominance values:

[0316] q=(32, 50, 50, 100, 100, 100, 180, 200, 200, 400, 720, 720, 1440).

[0317] However, in one embodiment, the quantization factor q is chosen adaptively for each distinct tile of the image, based on the density of image features in the tile. Referring to Figure 4, the entries of subbands are labeled LH_k, HL_k and HH_k as $u_{ij}^{(k)}$, $v_{ij}^{(k)}$ and $w_{ij}^{(k)}$, respectively.

[0318] Referring to Figure 12, the block classifier module computes for each transform layer (e.g., k=1, 2, 3, 4, 5) of the tile a set of block classification values, as follows:

[0319] $$U_k = \sum_{ij} \left| u_{ij}^{(k)} \right|$$

[0320] $$V_k = \sum_{ij} \left| v_{ij}^{(k)} \right|$$

[0321] $$W_k = \frac{1}{2} \sum_{ij} \left| w_{ij}^{(k)} \right|$$

[0322] $$B_k = \max\{U_k, V_k, W_k\}$$

[0323] $$S_k = \sqrt{\frac{1}{2}\left\{U_k^2 + V_k^2 + W_k^2 - \frac{1}{3}(U_k + V_k + W_k)\right\}}$$

[0324] Vertical and horizontal lines in the original image will mostly be represented by $u_{ij}^{(k)}$ and $v_{ij}^{(k)}$, respectively. B_k tends to be large if the original image (i.e., in the tile being evaluated by the block classifier) contains many features (e.g., edges and textures). Therefore, the larger the value of B_k, the harder it will be to compress the image without creating compression artifacts.

[0325] Using a two-class model, two quantization tables are provided:

[0326] Q0 = (16, 16, 16, 18, 18, 18, 36, 36, 36, 72, 72, 72 144. 300, 300, 600),

[0327] Qr - (16, 32, 32, 36, 36, 36, 72, 72, 72, 144, 144, 144, 288, 660, 600, 1200)

[0328] where Q_0 is used for "hard" to compress blocks and Q_1 is used for "easy" to compress blocks.

[0329] Interior tiles (i.e., tiles not on the boundary of the image) are each classified as either "hard" or "easy" to compress based on a comparison of one or more of the B_k values with one or more respective threshold values. For instance, as shown in Figure 12, B_1 for a tile may be compared with a first threshold TH1 (e.g., 65) (step 271). If B_1 is greater than the threshold, then the tile is classified as "hard" (step 272). Otherwise, B_5 is compared with a second threshold TH2 (e.g., 60) (step 273). If B_5 is greater than the second threshold, then the tile is classified as "hard" (step 274), and otherwise it is classified as "easy" (step 275). The wavelet coefficients for the tile are then quantized using the quantization divisors specified by the quantization table corresponding to the block (i.e., tile) classification.

[0330] In one embodiment, boundary tiles are classified by comparing B_1 with another, high threshold value TH1B, such as 85. Boundary tiles with a B_1 value above this threshold are classified as "hard" to compress and otherwise are classified as "easy" to compress.

[0331] In an alternate embodiment, three or more block classifications may be designated, and a corresponding set of threshold values may be defined.

Based on comparison of B_1, and/or other ones of the B_i values with these thresholds, a tile is classified into one of the designated classifications, and a corresponding quantization table is then selected so as to determine the quantization values to be applied to the subbands within the tile. S_k also tends to be large if the original image contains many features, and therefore in some embodiments k is used instead of B_k to classify image tiles.

[0332] *Sparse Data Encodingwith Division between Significant and Insignificant Portions*

[0333] Referring to Figures 13A and 13B, once the transform coefficients for a tile of base image have been generated and quantized, the next step is to encode the resulting coefficients of the tile. A group of computational steps 280 are repeated for each NQS subband. The bitstreams generated by encoding each NQS subband are divided by bit planes and then grouped together to form the bitstreams stored in the image Figures 8A to 8E.

[0334] Referring to Figure 13A, the encoding procedure or apparatus determines the maximum bit depth of the block of data in the NQS subband to be encoded (286), which is the maximum number of bits required to encode any of the coefficient values in the block, and is herein called the

maximum bit depth, or MaxbitDepth. The value of MaxbitDepth is determined by computing the maximum number of bits required to encode the absolute value of any data value in the block. In particular, MaxbitDepth is equal to int(log2V) + 1, where V is the largest absolute value of any element in the block, and "int()" represents the integer portion of a specified value. The maximum bit depth for each top level block is stored in a corresponding bitstream (e.g., the significant bitstream for the subband group whose coefficients are being encoded). Next, the Block procedure is invoked for the current block (288). A pseudocode representation of the block procedure is shown in Table 2.

[0335] Each block contains four subblocks (see Figure 14A). As shown in Figure 13B, the Block procedure determines the MaxbitDepth for each of the four subblocks of the current block (300). Then, it generates and encodes a MaxbitDepth mask (301). The mask has four bits: m_1, m_2, m_3 and m_4, each of which is set equal to a predefined value (e.g., 1) only if the MaxbitDepth of the corresponding subblock is equal to the MaxbitDepth m_0 of the current (parent) block, and is otherwise set to zero. The mathematical representation of the mask is as follows:

[0336] $mask=(m_0== m_1)+(m_0== m_2)+(m_0== m_3)+(m_0== m_4)$

[0337] where the "+" in the above equation represents concatenation.

[0338] For example, a mask of 1000 indicates that only subblock 1 has a MaxbitDepth equal to the MaxbitDepth of the current block. The value of the mask is between 1 and 15.

[0339] The MaxbitDepth mask is preferably encoded using a 15-symbol Huffman table (see Table 1). As shown, the four mask values that correspond to the most common mask patterns, where just one subblock having a MaxbitDepth equal to the MaxbitDepth of the parent block, are encoded with just three bits.

[0340] Table 1

Huffman Table for Encoding MaxbitDepth Mask

Mask	Huffman Code
0001	111
0010	101
0011	1001
0100	011
0101	0010
0110	10000
0111	01001
1000	110
1001	01000
1010	0001
1011	00110
1100	0101
1101	00111
1110	0000
1111	10001

[0341] *Encoding Subblock MaxbitDepth Values*

[0342] In addition, step 301 includes encoding the MaxbitDepth value for each of the subblocks whose MaxbitDepth is not equal to the MaxbitDepth m of the current block. For instance as shown in Figures 14A and 14B, if the MaxbitDepth values for the current block are

$$m_1, m_2, m_3, m_4 = 5, 0, 3, 2$$

then the only MaxbitDepth values that need to be encoded are m_2, m_3, m_4, because the MaxbitDepth value of m_1 is known from the MaxbitDepth mask and the previous stored and encoded value of the MaxbitDepth m_0 of the current block.

[0343] It should be noted that if $m_0 = 1$, then there is no need to encode the MaxbitDepth values of the subblocks, because those values are known completely from the MaxbitDepth mask.

[0344] If $m_0 \neq 1$, then for each $m_i \neq m_0$, the procedure encodes the value m_i as follows:

[0345] $m_i = 0$, then the procedure outputs a string of 0's of length m_0-1 ; and

[0346] otherwise, the procedure outputs a string of 0's of length $m_0 - m_i$-1 followed by a 1.

[0347] For instance, if m_0= 5 and m_1= 0, then m_1 is encoded as a string of four 0's: 0000. If m_0=5 and m_2=3, then m_2 is encoded as string of (5 - 3 - 1 = 1) one 0 followed by a 1 : 0 l.

[0348] In the example of {m_1, m_2, m_3, m_4} = {5, 0, 3, 2}, the MaxbitDepth values are encoded as follows:

mask	m_2 Subblock	m_3 Subblock	m_4 Subblock
111	0000	01	001

[0349] Next, if the coefficients of the NQS subband being encoded are to be stored in two or more bitstreams, then the encoded representation of the MaxbitDepth values for the block is divided into two more portions, with each portion containing the information content for a certain range of bit planes. For ease of explanation, an explanation in detail is provided as to how the MaxbitDepth values and mask and coefficient values are split between two portions, herein called the significant and insignificant portions. The same technique is used to split these values between three bit plane ranges corresponding significant, mid-significant and insignificant for least significant) portions.

[0350] For each NQS subband, excluding the last group of NQS subbands, the coefficient bit planes are divided into two or three ranges. When there

are two bit plane ranges, a bit plane threshold that divided the two ranges is chosen or predefined. The "insignificant" portion of each "coefficient value" (including its MaxbitDepth value) below the bit plane threshold is stored in an "insignificant" bitstream 206 (see Figure 8D), and the rest of the coefficient is stored in the corresponding significant bitstream 206. Selection of the bit plane ranges is typically done on an experimental basis, but encoding numerous images using various bit plane ranges, and then selecting a set of bit plane ranges that, on average, achieves specified division of data between the bitstreams for the various resolution levels. For example, the specified division may be an approximately equal division of data between the bitstream for a first resolution level and the next resolution level. Alternately, the specified division may call for the bitstreams for a second resolution level to contain four times as much data as the bitstreams for a first (lower) resolution level.

[0351] The splitting of MaxbitDepth values between significant and insignificant portions will be addressed initially, and then the encoding and splitting of coefficient values for minimum size blocks will be addressed.

[0352] If the MaxbitDepth m_0 of a block is less than the threshold, the MaxbitDepth mask and every bit of the MaxbitDepth values for the

subblocks are stored in the insignificant portion of the base image subfile. Otherwise, the MaxbitDepth mask is stored in the significant part, and then each of the encoded subblock MaxbitDepth values are split between significant and insignificant parts as follows. This splitting is handled as follows $m_i \geq$ threshold, the entire encoded MaxbitDepth value m_i is included in the significant portion of the subimage subfile. Otherwise, the first m_0 threshold bits of each MaxbitDepth value m_i, excluding $m_i = m_0$, are stored in the significant portion of the subimage subfile and the remaining bits of each m_i (if any) are stored in the insignificant portion of the subimage subfile.

[0353] If the bit planes of the coefficients are to be divided into three ranges, then two bit plane thresholds are chosen or predefined, and the MaxbitDepth mask and values are allocated among three bitstreams using the same technique as described above.

[0354] *Encoding Coefficient Values for Minimum Size Block*

[0355] Next, if the size of the current block (i.e., the number of coefficient values in the current block) is not a predefined minimum number (302-No), such as four, then the Block procedure is called for each of the four subblocks of the current block (303). This is a recursive procedure call. As a

result of calling the Block procedure on a subblock, the MaxbitDepth mask and values for the subblock are encoded and inserted into the pair of bitstreams for the subband group being encoded. If the subblock is not of the predefined minimum size, then the Block procedure is recursively called on its subblocks, and so on.

[0356] When a block of the predefined minimum size is processed by the block procedure (302-Yes), after the MaxbitDepth mask for the block and the MaxbitDepth values of the subblocks have been encoded (301), the coefficients of the block are encoded, and the encoded values are split between significant and insignificant parts (304).

[0357] Each coefficient that is not equal to zero includes a POS/NEG bit to indicate its sign, as well as a MaxbitDepth number of additional bits. Further, the MSB (most significant bit) of each non-zero coefficient, other than the sign bit, is already known from the MaxbitDepth value for the coefficient, and in fact is known to be equal to 1. Therefore, this MSB does not need to be encoded (or from another viewpoint, it has already been encoded with the MaxbitDepth value).

[0358] For each coefficient of a minimum size block, if the MaxbitDepth of the coefficient is less than the threshold, then all the bits of the coefficient,

including its sign bit, are in the insignificant portion. Otherwise, the sign bit is in the significant portion, and furthermore the most significant bits (MSG's), if any, above the threshold number of least significant bits (LSB's), are also included in the significant portion. In other words, the bottom "threshold" number of bits are allocated to the insignificant portion. However, if the MaxbitDepth is equal to the threshold, the sign bit is nevertheless allocated to the significant portion and the remaining bits are allocated to the insignificant portion.

[0359] Furthermore, as noted above, since the MSE of the absolute value of each coefficient is already known from the MaxbitDepth mask and values, that bit is not stored. Also, coefficients with a value of zero are not encoded because their value is fully known from the MaxbitDepth value of the coefficient, which is zero.

[0360] For example (see Figure 14C), consider four coefficients {31, 0, -5, -2} of a block whose values are with binary values are POS 11111, 0, NEG 101, NEG 10, and a threshold value of 3. First the zero value coefficients and the MSB's of the non-zero coefficient are eliminated to yield: POS 1111, NEG 01, NEG 0. Then the threshold number of least significant bits (other than sign

bits) are allocated to the insignificant portion and the rest are allocated to the significant portion as follows:

[0361] significant portion: POS 1, NEG

[0362] insignificant portion: 111, 01, NEG 0.

[0363] The significant portion contains the most significant bits of the 31 and –5 coefficient values, while the insignificant portion contains the remaining bits of the 31 and –5 coefficient values and all the bits of the –2 coefficient value.

[0364] Table 2

Pseudocode for Block Encoding Procedure

```
//Encode MaxbitDepth m_i for each subblock i:
Determine MaxbitDepth m_i for each subblock i =1, 2, 3, 4
mask=(m_0==m_1)+( m_0==m_2)+( m_0==m_3)+( m_0==m_4)
//where the "+" in the above equation represents concatenation
Encode and store mask using Huffman table

For i=1 to 4{
   If m_i≠m_0 {
      if mi=0 {
         output a string of m_0 0's }
      else {        //m_i≠0
         output a string of m_0-m_i 0's, followed by a 1 }
      }
   }
```

```
//  Divide the encoded MaxbitDepth mask and MaxbitDepth between
// significant and insignificant portions as follows:
If m0<threshold {
   output the MaxbitDepth mask and MaxbitDepth values to insignificant
bitstream }
else {
   output the MaxbitDepth mask to significant bitstream;
   for i = 1 to 4 {
      if mi=m0 {output nothing for that mi}
      else {
         if mi≥ threshold { output mi to significant bitstream }
         else  {
            output the first m0-threshold bits of mi to the significant bitstream
            and output the remaining bits of mi (if any) in the insignificant
            bitstream  }
         }
      }
   }

// Encode Coefficient values if block is of minimum size
If size of current block is > minimum block size {
   // coefficient values are denoted as ci
   for i = 1 to 4 {
      Call Block(subblock i);
      }
   else {          // size of current block is ≤minimum block size
      C = number of coefficients in block;    // if block size is already known,
      skip this step for i=1 to C {
         if mi<threshold  {
            output all bits of ci to insignificant bitstream;
            }
         else   {
            output sign(ci) to the significant bitstream;
            if mi > threshold  {
               #M = mi – threshold – 1;  // #M ≥ 0
               output the #M most significant bits to the significant bitstream;
               }
```

output all remaining least significant bits of c_i to the insignificant bitstream;

}

} // end of coefficient processing loop

} // end of main else clause

} // end of procedure

Return

[0365] As discussed above, if the bit planes of the coefficients are to be divided into three ranges, then two bit plane thresholds are chosen or predefined, and the encoded coefficient values are allocated among three bitstreams using the same technique as described above.

[0366] *Image Reconstruction*

[0367] To reconstruct an image from an image file, at a specified resolution level that is equal to or lower than the resolution level at which the base image in the file was encoded, each bitstream of the image file up to the specified resolution level is decompressed and dequantized. Then, on a tile by tile basis the reconstructed transform coefficients are inverse transformed to reconstruct the image data at specified resolution level.

[0368] Referring to Figure 15 the image reconstruction process reconstructs an image from image data received from an image file (320). A user of the procedure or device performing the image reconstruction, or a control

procedure operating on behalf of a user, selects or specifies a resolution level R that is equal to or less than the highest resolution level included in the image data (322). A header of the image data file is read to determine the number and arrangement of tiles (L, K) in the image, and other information that may be needed by the image reconstruction procedure (323). Steps 324 and 326 reconstruct the image at the given resolution level, and at step 328 the reconstructed image is displayed or stored in a memory device. Figures 16A and 16B provide a more detailed view of the procedure for decoding the data for a particular tile at a particular subimage level.

[0369] In one embodiment, as shown in Figure 15, the data in the image file relevant to the specified resolution level is initially reorganized into tile by tile subfiles, with each tile subfile containing the bitstreams for that tile (324). Then, the data for each tile is processed (326). The header information is read to determine the MaxbitDepth for each top level subband block of the tile, the quantization factor used to quantize each subimage subband, and the like. The transform coefficients for each NQS subband required to reconstruct the image at the specified resolution level are decoded, in subband order. The details of the decoding process for decoding the coefficients in any one NQS subband are discussed below with reference to

Figure 16B. The resulting decoded coefficients are de-quantized applying the quantization factors for each subband (obtained from the Q table identified in the base image header). Then an inverse transform is applied to the resulting de-quantized coefficients. Note that the wavelet-like inverse transforms for reconstructing an image from the dequantized transform coefficients have been described above.

[0370] Referring to Figure 16A, to decode the data for one tile t at a specified resolution level, a set of steps 340 are repeated to decode each NQS subband of the tile, excluding those NQS subbands not needed for the specified resolution level and also excluding any bitstreams containing bit planes of encoded coefficient values not needed for the specified resolution level. Referring to Figures 8D and 8E, only the bitstreams of the base image needed to the specified resolution level are decoded. For a particular top level block (corresponding to a NQS subband) of the tile being decoded, the MaxbitDepth of the top level block is determined from either the header of the tile array (if the data has been reorganized into tile arrays) or from the data at the beginning of the bitstream(S) for the subband (346), and then the Decode-Block procedure is called to decode the data for the current block (348).

[0371] After the data for a particular subband has been decodeed, the decoded transform coefficients for that subband may be de-quantized, applying the respective quantization factor for the respective (350). Alternately, de-quantization can be performed after all coefficients for all the subband have been decoded.

[0372] Once all the coefficients for the NQS subbands have been decoded and de-quantized, an inverse transform is performed so as to regenerate the image data for the current tile t at the specified resolution level (352).

[0373] In an alternate embodiment, step 324 of Figure 15 is not used and the data in the image file is not reorganized into tile arrays. Rather, the image data is processed on a subband group by subband group basis, requiring the recovered transform coefficients for all the tiles to be accumulated and stored during the initial reconstruction steps. The steps 340 for decoding the data for one top level block of a particular tile for a particular subband group are repeated for each tile. In particular, for a particular top level block of a particular tile of a particular subband group, the MaxbitDepth of the top level block is determined from either the header of the tile array or from the data at the beginning of the bitstream(s) for the subband group (346), and

then the Decode-Block procedure is called to decode the data for the current block (348).

[0374] Referring to Figure 16B, the Decode-Block procedure (which is applicable to both the preferred and alternate embodiments mentioned in the preceding paragraphs) begins by decoding the MaxbitDepth data in the applicable encoded data array so as to determine the MaxbitDepth of each subblock of the current block (360). Depending on the NQS subband being decoded, the MaxbitDepth data for a block may be in one bitstream or may be split between two or three bitstreams, as described above, and therefore the applicable MaxbitDepth data bits from all required bitstreams will be read and decoded. If the size of the current block is greater than a predefined minimum block size (362-No), then the Decode-Block procedure is called for each of the subblocks of the current block (363). This is a recursive procedure call. As a result of calling the Decode-Block procedure on a subblock, the MaxbitDepth values for the subblock are decoded. If that subblock is not of the predefined minimum size, then the Decode-Block procedure is recursively called on its subblocks, and so on. When a block of the predefined minimum size is processed by the Decode-Block procedure (362-Yes), the coefficients of the block are decoded. Depending on the

subband group being decoded, the encoded coefficients for a block may be in one bitstream or maybe split between two or three bitstreams, as described above, and therefore the applicable, data bits from all required bitstreams will be read and decoded. Referring to Figure 16A, the quantized transform coefficients for each tile are regenerated for all NQS subbands included in the specified resolution level. After these coefficients have been de-quantized, the inverse transform is applied to each tile (352), as already described.

[0375] *Embodiment Using Non-Alternating Horizontal and Vertical Transforms*

[0376] In another embodiment, each tile of the image is first processed by multiple (e.g., five) horizontal decomposition transform layers and then by a similar number of vertical decomposition transform layers. Equivalently, the vertical transform layers could be applied before the horizontal transform layers. In hardware implementations of the image transformation methodology described herein, this change in the order of the transform layers has the advantage of either (A) reducing the number of times the data array is rotated, or (B) avoiding the need for circuitry that switches the roles of rows and columns in the working image array(s). When performing

successive horizontal transforms, the second horizontal transform is applied to the leftmost array of low frequency coefficients generated by the first horizontal transform, and the third horizontal transform is applied to the leftmost array of low frequency coefficients generated by the second horizontal transform, and so on. Thus, the second through Nth horizontal transforms are applied to twice as much day as in the transform method in which the horizontal and vertical transforms alternate. However, this extra data processing generally does not take any additional processing time in hardware implementations because in such implementations the horizontal filter is applied simultaneously to all rows of the working image array. The vertical transforms are applied in succession to successively smaller subarrays of the working image array. After the image data has been transformed by all the transform layers to (both horizontal and vertical), the quantization and encoding steps described above are applied to the resulting transform coefficients to complete the image encoding process.

[0377] As explained above, different (and typically shorter) transform filters may be applied to coefficients near the edges of the arrays being processed than the (typically longer) transform filter applied to coefficients away from those array edges. The use of longer transform filters in the middle provides

better data compression than the shorter transform filters, while the shorter transform filters eliminate the need for data and coefficients from neighboring tiles.

[0378] *Digital Camera Architecture*

[0379] Referring to Figure 17, there is shown an embodiment of a digital camera system 400. The digital camera system 400 includes an image capture device 402, such as a CCD or CMOS sensor array or any other mechanism suitable for capturing an image as an array of digitally encoded information. The image capture device is assumed to include analog to digital conversion (ADC) circuitry for converting analog image information into digital values. A working memory 404, typically random access memory, receives digitally encoded image information from the image capture device 402. More generally, it is used to store a digitally encoded image while the image is being transformed and compressed and otherwise processed by the camera's data (i.e., image) processing circuitry 406. In one embodiment, the data processing circuitry 406 consists of hardwired logic and a set of state machines for performing a set of predefined image processing operations.

[0380] In alternate embodiments, the data processing circuitry 406 could be implemented in part or entirely using a fast general purpose microprocessor and a set of software procedures. However, at least using the technology available in 2000, it would be difficult to process and store full resolution images (e.g., full color images having 1280 x 840 pixels) fast enough to enable the camera to be able to take, say, 20 pictures per second, which is a requirement for some commercial products. If, through the use of parallel processing techniques or well designed software, a low power, general purpose image data microprocessor could support the fast image processing needed by digital cameras, then the data processing circuit 106 could be implemented using such a general purpose microprocessor.

[0381] Each image, after it has been processed by the data processing circuitry 406, is typically stored as an "image file" in a nonvolatile memory storage device 408, typically implemented using "flash" (i.e., EEPROM) memory technology. The nonvolatile memory storage device 408 is preferably implemented as a removable memory card. This allows the camera's user to remove one memory card, plug in another, and then take additional pictures. However, in some implementations, the nonvolatile memory storage device 408 may not be removable, in which case the camera

will typically have a data access port 410 to enable the camera to transfer image files to and from other devices, such as general purpose, desktop computers.

[0382] Digital cameras with removable nonvolatile memory 408 may also include a data access port. The digital camera 400 includes a set of buttons 412 for giving commands to the camera. In addition to the image capture button, there will typically be several other buttons to enable the use to select the quality level of the next picture to be taken, to scroll through the images in memory for viewing on the camera's image viewer 414, to delete images from the nonvolatile image memory 408, and to invoke all the camera's other functions. Such other functions might include enabling the use of a flash light source, and transferring image files to and from a computer. In one embodiment, the buttons are electromechanical contact switches, but in other embodiments at least some of the buttons may be implemented as touch screen buttons on a user interface display 416, or on the image viewer 414.

[0383] The user interface display 416 is typically implemented either (A) as an LCD display device separate from the image viewer 414, or (B) as images displayed on the image viewer 414. Menus, user prompts, and information

about the images stored in the nonvolatile image memory 108 may be displayed on the user interface display 416, regardless of how that display is implemented.

[0384] After an image has been captured, processed and stored in nonvolatile image memory 408, the associated image file may be retrieved from the memory 408 for viewing on the image viewer. More specifically, the image tile is converted from its transformed, compressed form back into a data array suitable for storage in a framebuffer 418. The image data in the framebuffer is displayed on the image viewer 414. A date/time circuit 420 is used to keep track of the current date and time, and each stored image is date stamped with the date and time that the image was taken.

[0385] Still referring to Figure 17, the digital camera 400 preferably includes data processing circuitry for performing a predefined set of primitive operations, such as performing, the multiply and addition operations required to apply a transform to a certain amount of image data as well as a set of state machines 430-442 for controlling the data processing circuitry so as to perform a set of predefined image handling operations. In one embodiment, the state machines in the digital camera are as follows:

[0386] One or more state machines 430 for transforming, compressing and storing an image received from the camera's image capture mechanism. This image is sometimes tilled the "viewfinder" image, since the image being processed is generally the one seen, on the camera's image viewer 414. This set of state machines 430 are the ones that each image file stored in the nonvolatile image memory 408. Prior to taking the picture, the user specifies the quality level of the image to be stored using the camera's buttons 412. In one embodiment, the image encoding state machines 430 implement one or more features described above.

[0387] One or more state machines 432 for decompressing, inverse transforming and displaying a stored image tile on the camera's image viewer. The reconstructed image generated by decompressing, inverse transforming and dequantizing the image data is stored in camera's framebuffer 418 so that it can be viewed on the image viewer 414.

[0388] One or more state machines 434 for updating and displaying a count of the number of images stored in the nonvolatile image memory 408. The image count is preferably displayed on the user

interface display 416. This set of state machines 434 will also typically indicate what percentage of the nonvolatile image memory 408 remains unoccupied by image files, or some other indication of the camera's ability to store additional images. If the camera does not have a separate interface display 416, this memory status information may be shown on the image viewer 414, for instance superimposed on the image shown in the image viewer 414 or shown in a region of the viewer 414 separate from the main viewer image.

[0389] One or more state machines 436 for implementing a "viewfinder" mode for the camera in which the image currently "seen" by the image capture mechanism 402 is displayed on the image viewer 414 so that the user can see the image that would be stored if the image capture button is pressed. These state machines transfer the image received from the image capture device 402, possibly after appropriate remedial processing steps are performed to improve the raw image data, to the camera's framebuffer 418.

[0390] One or more state machines 438 for downloading images from the nonvolatile image memory 408 to an external device, such as a general purpose computer (one or more state machines 440 for

uploading images from an external device, such as a general purpose computer, into the nonvolatile image memory 408. This enables the camera to be used as an image viewing device, and also as a mechanism for transferring image files on memory cards.

[0391] *Alternate Embodiments*

[0392] Generally, the present invention is useful in any "memory conservative" context where the amount of working memory available is insufficient to process entire images as a single tile, or where a product must work in a variety of environments including low memory environments, or where an image may need to be conveyed over a low bandwidth communication channel or where it may be necessary or convenient to providing image at a variety of resolution levels.

[0393] In streaming data implementations, such as in a web browser that receives compressed images encoded using the present invention, subimages of an image may be decoded and decompressed on the fly, as the data for other higher level subimages of the image are being received. As a result, one or more lower resolution versions of the compressed image may be reconstructed and displayed before the data for the highest resolution

version of the image is received (and/or decoded) over a communication channel.

[0394] In another alternate embodiment, a different transform than the wavelet-like transform described above could be used.

[0395] In alternate embodiments, the image tiles could be processed in a different order. For instance, the image tiles could be processed from right to left instead of left to right. Similarly, image tiles could be processed starting at the bottom row and proceeding toward the top row.

[0396] The present invention can be implemented as a computer program product that includes a computer program mechanism embedded in a computer readable storage medium. For instance, the computer program product could contain the program modules shown in Figure 5. These program modules may be stored on a CD-ROM, magnetic disk storage product, or any other computer readable data or program storage product. The software modules in the computer program product may also be distributed electronically, via the Internet or otherwise, by transmission of a computer data signal (in which the software modules are embedded) on a carrier wave.

[0397] While the present invention has been described with reference to a few specific embodiments, the description is illustrative of the invention and is not to be construed as limiting the invention. Various modifications may occur to those skilled in the art without departing from the true spirit and scope of the invention as defined by the appended claims.

[0398] Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.